CONVEX HULLS IN 3D

Vera Sacristán
Rodrigo Silveira

Discrete and Algorithmic Geometry
Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya

CONVEX HULLS IN 3D

STORING A CONVEX POLYHEDRON

CONVEX HULLS IN 3D

STORING A CONVEX POLYHEDRON

Planarity: Any convex polyhedron is a planar graph.

STORING A CONVEX POLYHEDRON

Planarity: Any convex polyhedron is a planar graph.

Proof: Consider a convex polyhedron P. Let p be a point in the interior of P and S be a sphere containing P. Due to its convexity, the central projection of P from p onto S is a graph G_1 isomorphic to P. Let q be a point in S interior to a face of G_1 and π be the plane tangent to S in the point diametrically opposed to q. The stereographic projection of S from q onto π maps G_1 into a graph G_2 which is isomorphic to G_1 and plane. Therefore, P is a planar graph.

CONVEX HULLS IN 3D

STORING A CONVEX POLYHEDRON

Planarity: Any convex polyhedron is a planar graph.

Proof: Consider a convex polyhedron P. Let p be a point in the interior of P and S be a sphere containing P. Due to its convexity, the central projection of P from p onto S is a graph G_1 isomorphic to P. Let q be a point in S interior to a face of G_1 and π be the plane tangent to S in the point diametrically opposed to q. The stereographic projection of S from q onto π maps G_1 into a graph G_2 which is isomorphic to G_1 and plane. Therefore, P is a planar graph.

Euler's relation: If G is a connected planar graph with v vertices, e edges, and f faces, $v + f = e + 2$.

CONVEX HULLS IN 3D

STORING A CONVEX POLYHEDRON

Planarity: Any convex polyhedron is a planar graph.

Proof: Consider a convex polyhedron P. Let p be a point in the interior of P and S be a sphere containing P. Due to its convexity, the central projection of P from p onto S is a graph G_1 isomorphic to P. Let q be a point in S interior to a face of G_1 and π be the plane tangent to S in the point diametrically opposed to q. The stereographic projection of S from q onto π maps G_1 into a graph G_2 which is isomorphic to G_1 and plane. Therefore, P is a planar graph.

Euler's relation: If G is a connected planar graph with v vertices, e edges, and f faces, $v + f = e + 2$.

Complexity: Any convex polyhedron with with n vertices has at most $3n - 6$ vertices and $2n - 4$ faces.

CONVEX HULLS IN 3D

STORING A CONVEX POLYHEDRON

Planarity: Any convex polyhedron is a planar graph.

Proof: Consider a convex polyhedron P. Let p be a point in the interior of P and S be a sphere containing P. Due to its convexity, the central projection of P from p onto S is a graph G_1 isomorphic to P. Let q be a point in S interior to a face of G_1 and π be the plane tangent to S in the point diametrically opposed to q. The stereographic projection of S from q onto π maps G_1 into a graph G_2 which is isomorphic to G_1 and plane. Therefore, P is a planar graph.

Euler's relation: If G is a connected planar graph with v vertices, e edges, and f faces, $v + f = e + 2$.

Complexity: Any convex polyhedron with with n vertices has at most $3n - 6$ vertices and $2n - 4$ faces.

Corollary: The convex hull of n points in E^3 is a convex polyhedron which can be stored using $O(n)$ space

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.

CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.

Theorem: The algorithm computes the convex hull of n points in E^3 in $O(n \log n)$ time and $O(n)$ space. These bounds are optimal.

CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.

$O(n)$ time

Theorem: The algorithm computes the convex hull of n points in E^3 in $O(n \log n)$ time and $O(n)$ space. These bounds are optimal.

CONVEX HULLS IN 3D

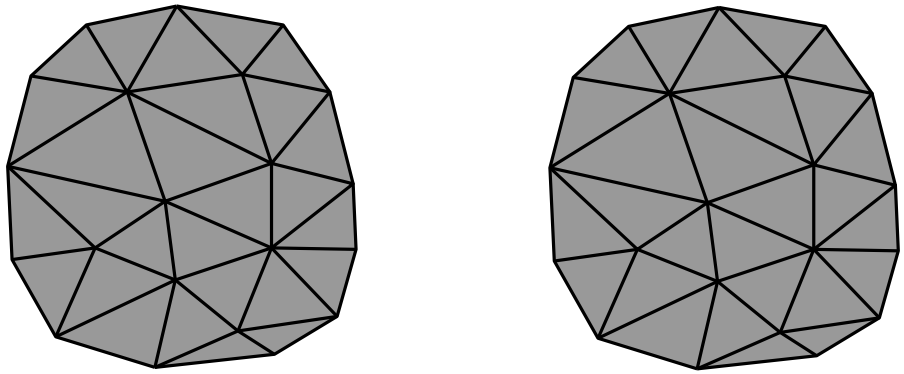
COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.

$O(n)$ time



Theorem: The algorithm computes the convex hull of n points in E^3 in $O(n \log n)$ time and $O(n)$ space. These bounds are optimal.

CONVEX HULLS IN 3D

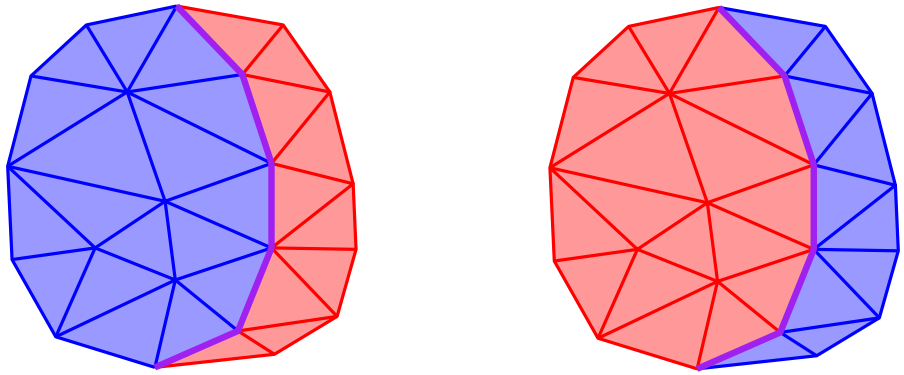
COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.

$O(n)$ time



Theorem: The algorithm computes the convex hull of n points in E^3 in $O(n \log n)$ time and $O(n)$ space. These bounds are optimal.

CONVEX HULLS IN 3D

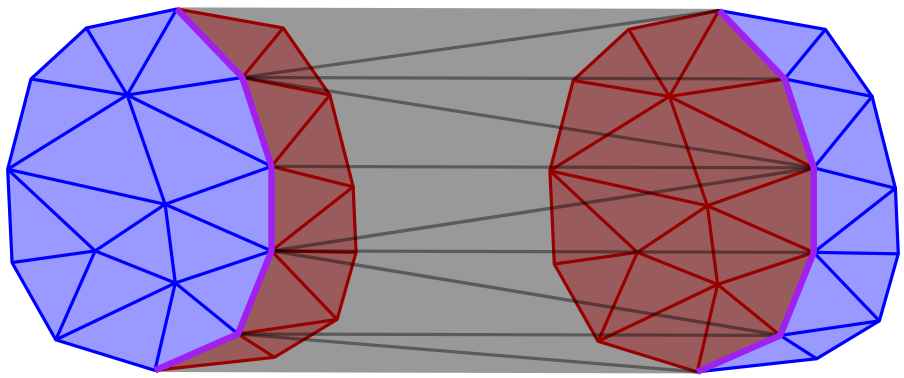
COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.

$O(n)$ time

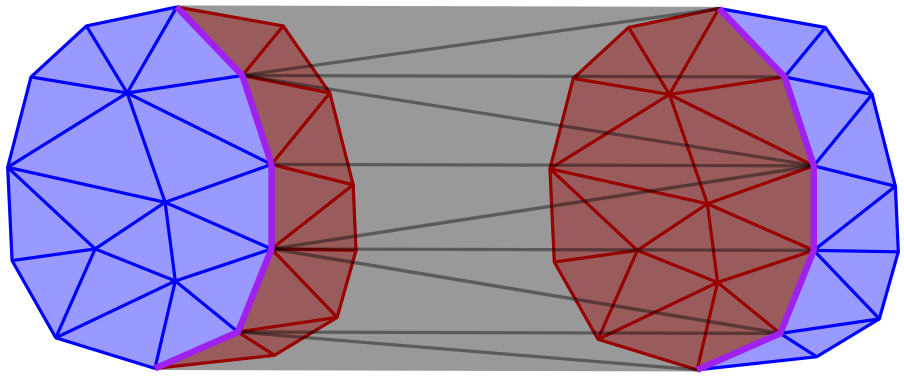


Theorem: The algorithm computes the convex hull of n points in E^3 in $O(n \log n)$ time and $O(n)$ space. These bounds are optimal.

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue



CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue

Be careful!

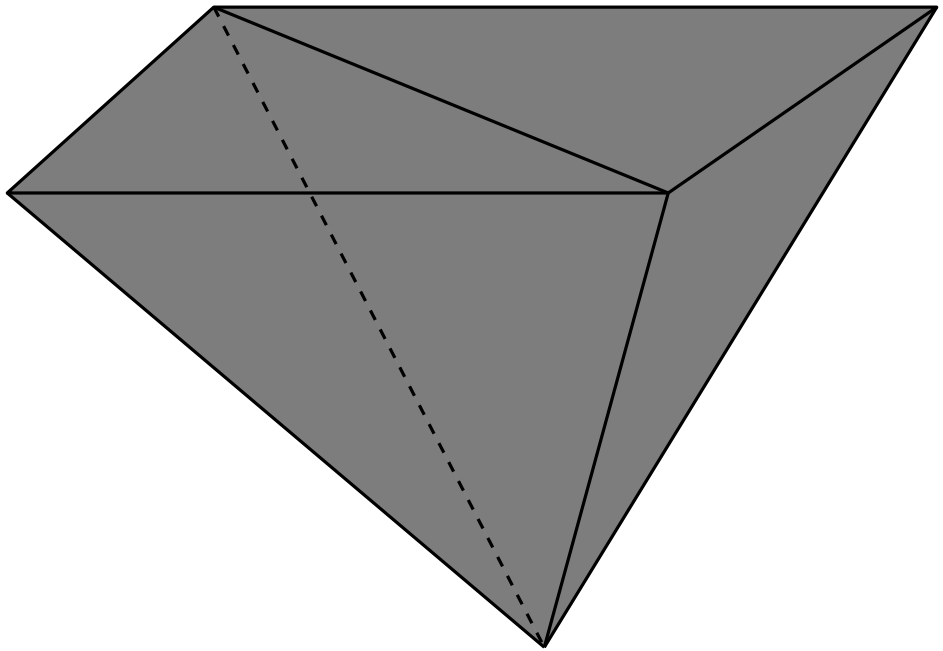
CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue

Be careful!



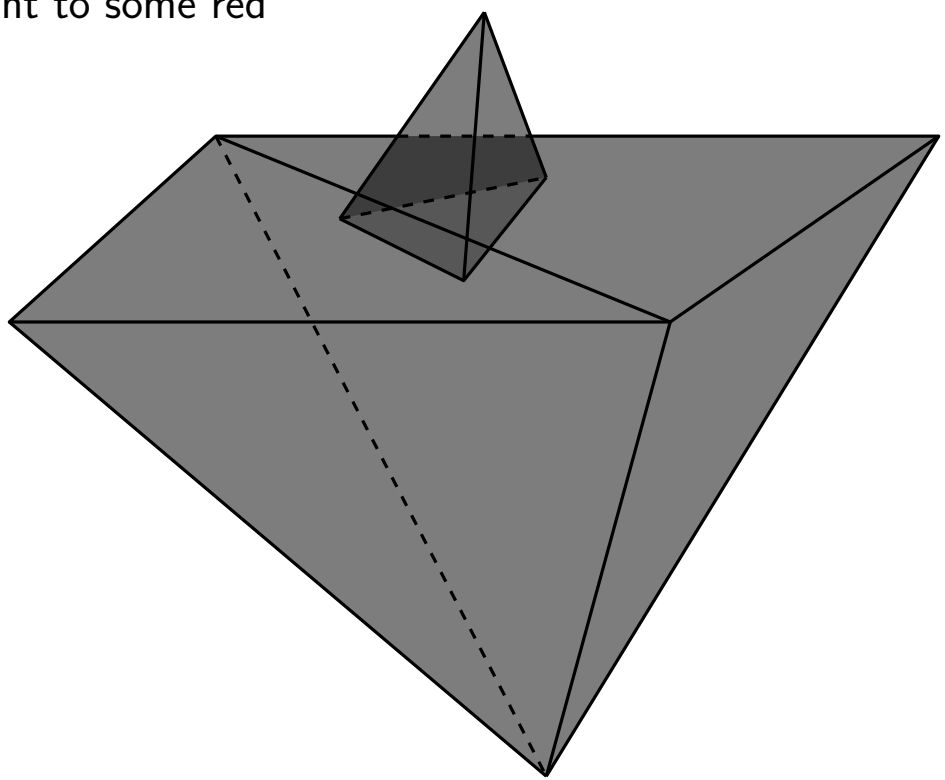
CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue

Be careful!

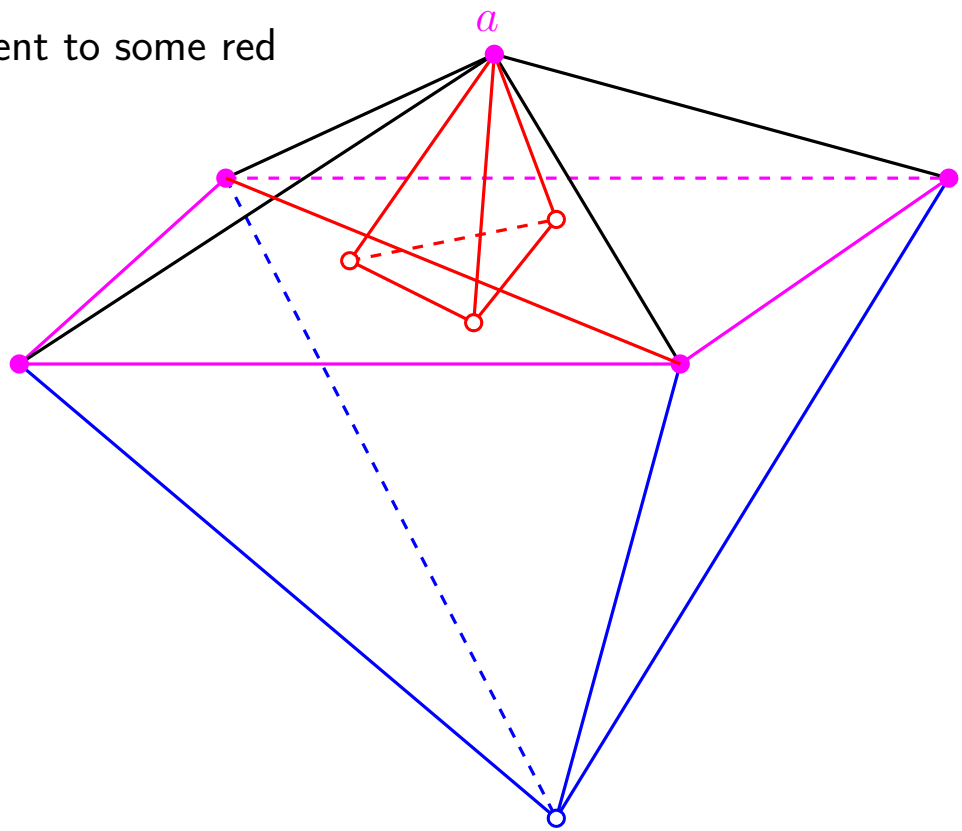


CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue



Be careful!

Vertex a is not incident to any purple edge.

Purple is not connected.

CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

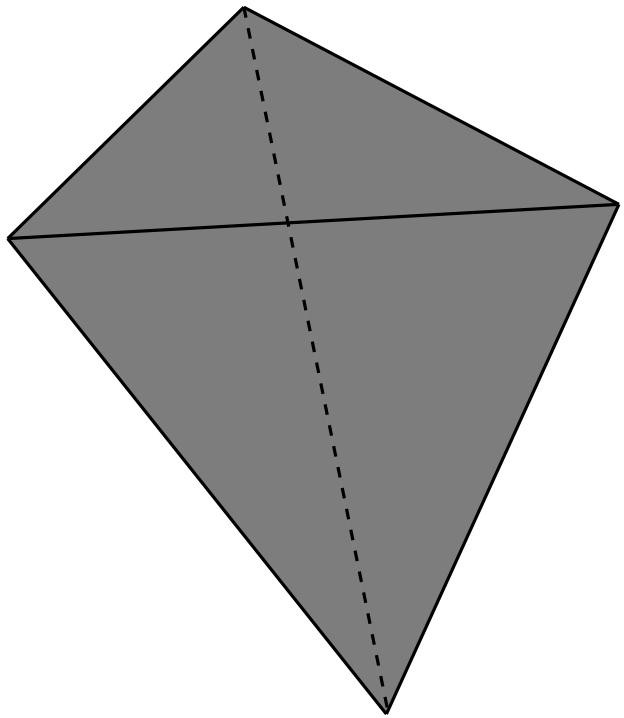
Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue

Be careful!

Vertex a is not incident to any purple edge.

Purple is not connected.

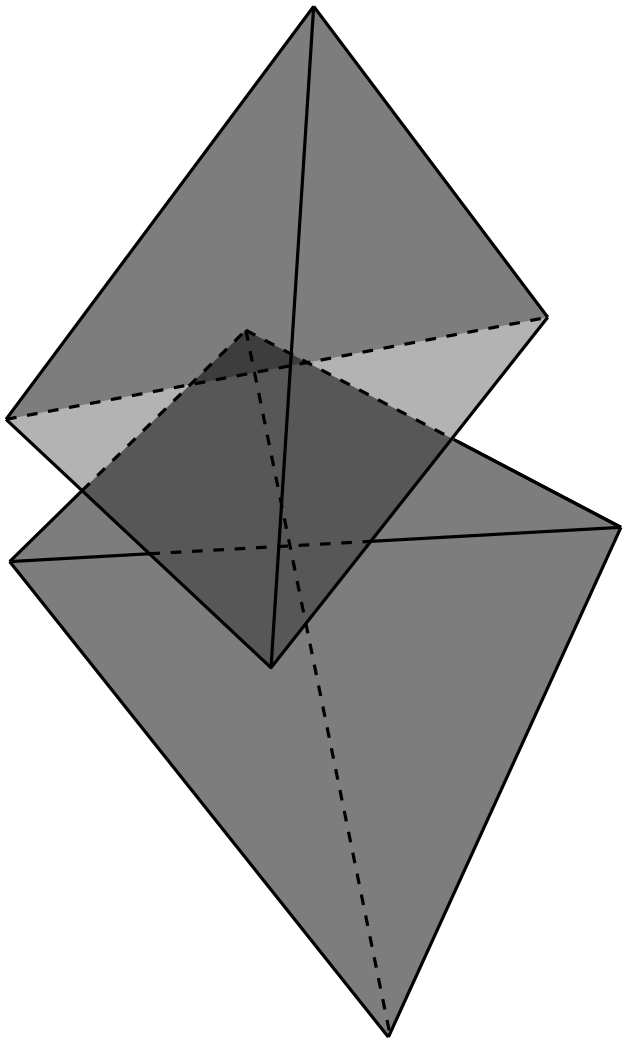


CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue



Be careful!

Vertex a is not incident to any purple edge.

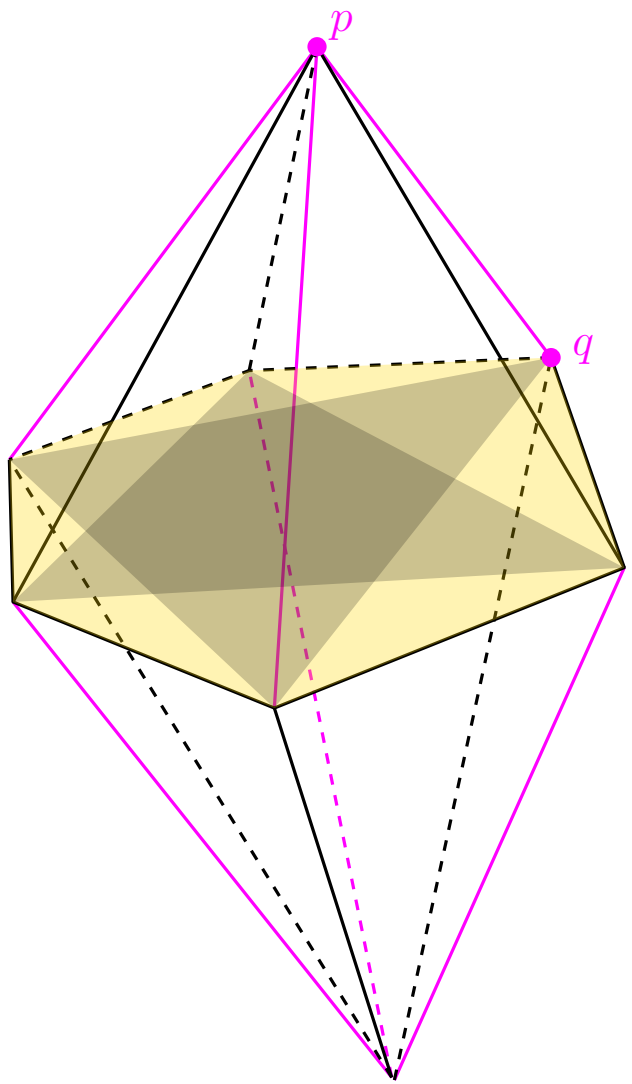
Purple is not connected.

CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue



Be careful!

Vertex a is not incident to any purple edge.

Purple is not connected.

Vertex p is incident to more than 2 purple edges

Edge pq is incident to two red faces

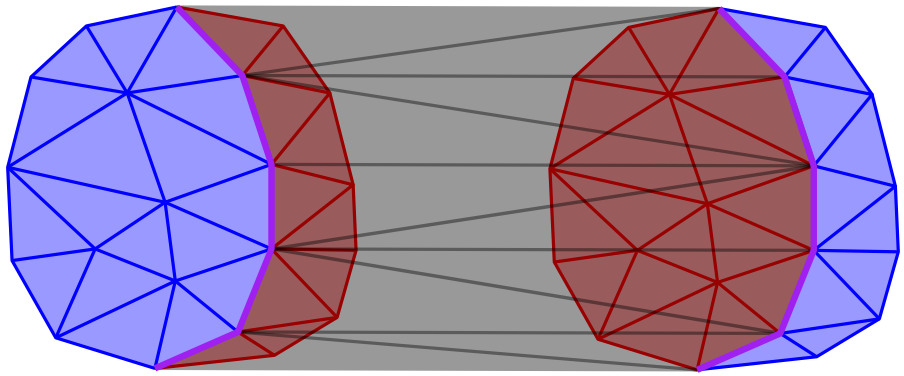
There are no blue faces

CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D BY DIVIDE AND CONQUER

Merge step (i.e., $C = ch(C_1 \cup C_2)$) – some notation

- Faces (i.e., facets=triangles, edges, or vertices) of C_1 or C_2 that are not in C are red
- Facets (i.e., triangles) of C_1 or C_2 that appear in C are blue
- Edges of C_1 or C_2 that appear in C: if incident to some red triangle, it is purple, otherwise it is blue
- Vertices of C_1 or C_2 that appear in C: if incident to some red or purple edge it is purple), otherwise it is blue



Be careful!

Vertex a is not incident to any purple edge.

Purple is not connected.

Vertex p is incident to more than 2 purple edges

Edge pq is incident to two red faces

There are no blue faces

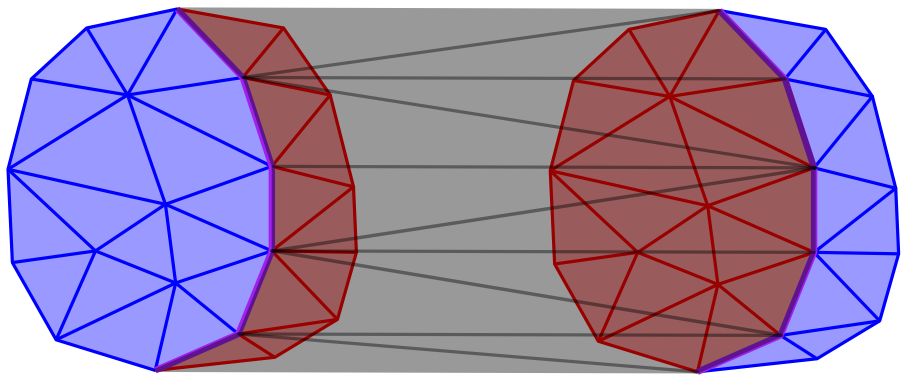
Corollary

It is not always true that the purple edges and vertices form a cycle.

COMPUTING A CONVEX HULL IN 3D

Observation 1 (new edges and new triangles)

The edges of $C \setminus C_1 \cup C_2$ are the convex hull of two purple vertices, one from C_1 and the other from C_2.
The triangles of $C \setminus C_1 \cup C_2$ are the convex hull of a purple vertex from C_i and a purple edge from C_j, $i \neq j$.



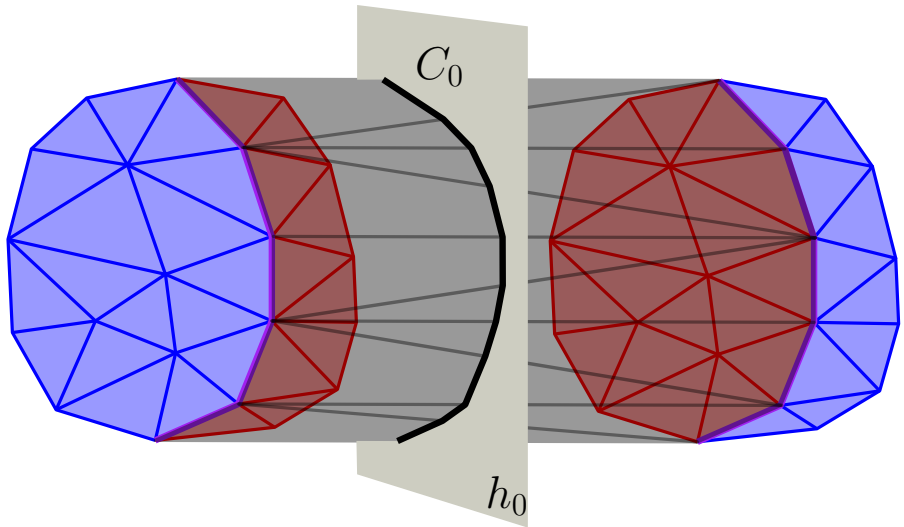
COMPUTING A CONVEX HULL IN 3D

Observation 1 (new edges and new triangles)

The edges of $C \setminus C_1 \cup C_2$ are the convex hull of two purple vertices, one from C_1 and the other from C_2.
The triangles of $C \setminus C_1 \cup C_2$ are the convex hull of a purple vertex from C_i and a purple edge from C_j, $i \neq j$.

Observation 2

Let h_0 be a plane separating C_1 from C_2. The faces and edges of $C \setminus C_1 \cup C_2$ intersect h_0 in a convex polygon C_0, and the circular order on the edges and vertices of C_0 induces a circular order on the faces and edges of $C \setminus C_1 \cup C_2$.



(plane $x = 0$)

CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D

Observation 1 (new edges and new triangles)

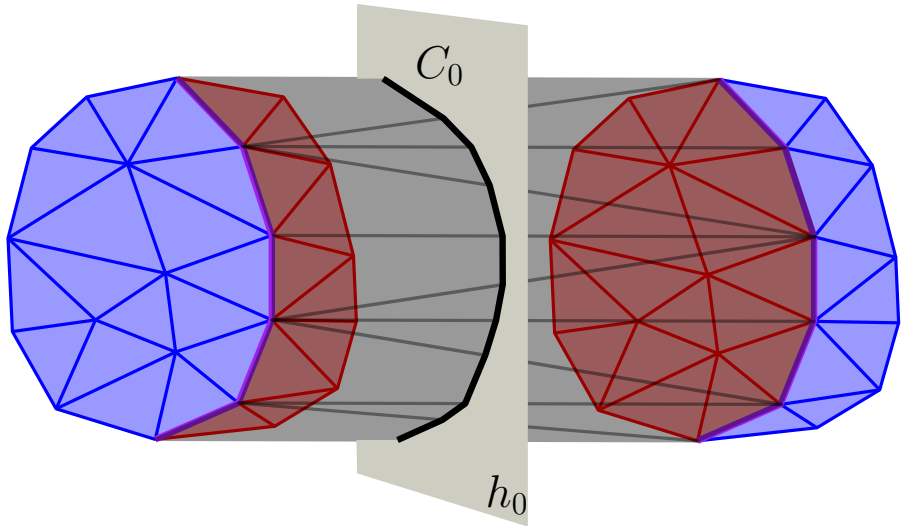
The edges of $C \setminus C_1 \cup C_2$ are the convex hull of two purple vertices, one from C_1 and the other from C_2.
The triangles of $C \setminus C_1 \cup C_2$ are the convex hull of a purple vertex from C_i and a purple edge from C_j, $i \neq j$.

Observation 2

Let h_0 be a plane separating C_1 from C_2. The faces and edges of $C \setminus C_1 \cup C_2$ intersect h_0 in a convex polygon C_0, and the circular order on the edges and vertices of C_0 induces a circular order on the faces and edges of $C \setminus C_1 \cup C_2$.

Merging algorithm

1. Find a first edge of $C \setminus C_1 \cup C_2$
2. Find the remaining new faces and edges in the order induced by C_0
3. Identify the red faces, edges and vertices and update the DCEL



(plane $x = 0$)

CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D

Observation 1 (new edges and new triangles)

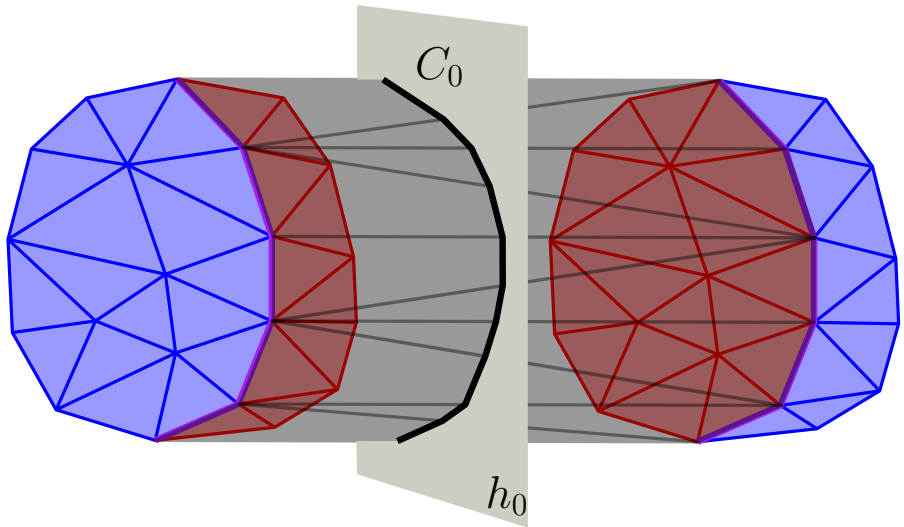
The edges of $C \setminus C_1 \cup C_2$ are the convex hull of two purple vertices, one from C_1 and the other from C_2.
The triangles of $C \setminus C_1 \cup C_2$ are the convex hull of a purple vertex from C_i and a purple edge from C_j, $i \neq j$.

Observation 2

Let h_0 be a plane separating C_1 from C_2. The faces and edges of $C \setminus C_1 \cup C_2$ intersect h_0 in a convex polygon C_0, and the circular order on the edges and vertices of C_0 induces a circular order on the faces and edges of $C \setminus C_1 \cup C_2$.

Merging algorithm

1. Find a first edge of $C \setminus C_1 \cup C_2$
2. Find the remaining new faces and edges in the order induced by C_0
3. Identify the red faces, edges and vertices and update the DCEL



(plane $x = 0$)

Important: recall that we do not know the purple vertices and edges in advance, we need to discover them on-the-fly

CONVEX HULLS IN 3D

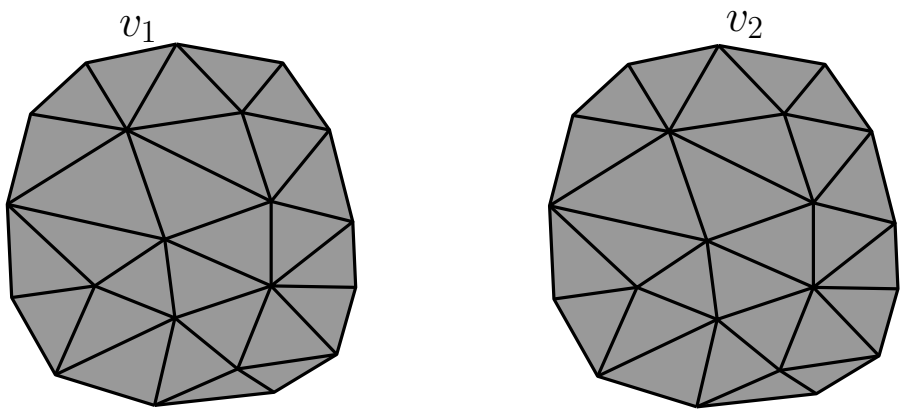
COMPUTING A CONVEX HULL IN 3D

Step 1: Finding the first new edge

COMPUTING A CONVEX HULL IN 3D

Step 1: Finding the first new edge

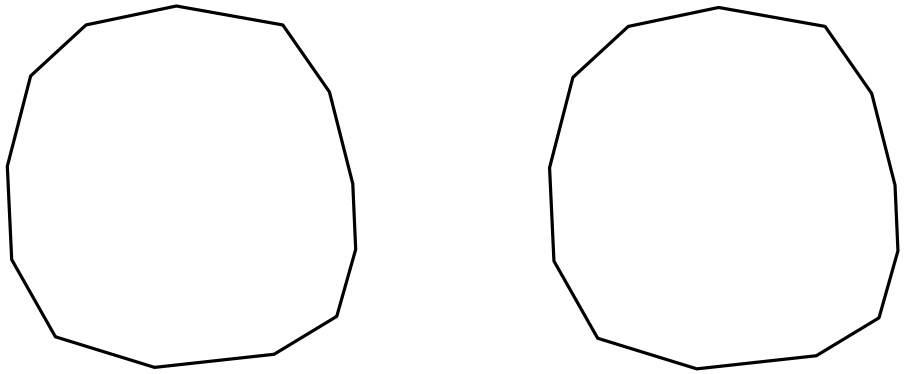
1. Orthogonally project C_1 and C_2 onto the plane $z = 0$. Let C_1' and C_2' respectively be their projections. The vertex v of greatest/smallest abscissa in C_i projects onto the vertex v' of greatest/smallest abscissa in C_i', so those are starting points to trace the boundaries of C_1' and C_2'
   - The following vertex of C_i' is the projection of one of the neigbors of v in C_i.
2. Find $v_1'v_2'$, one of the external bitangents of C_1' and C_2'.
3. Then v_1v_2 is an edge of $C \setminus C_1 \cup C_2$.



COMPUTING A CONVEX HULL IN 3D

Step 1: Finding the first new edge

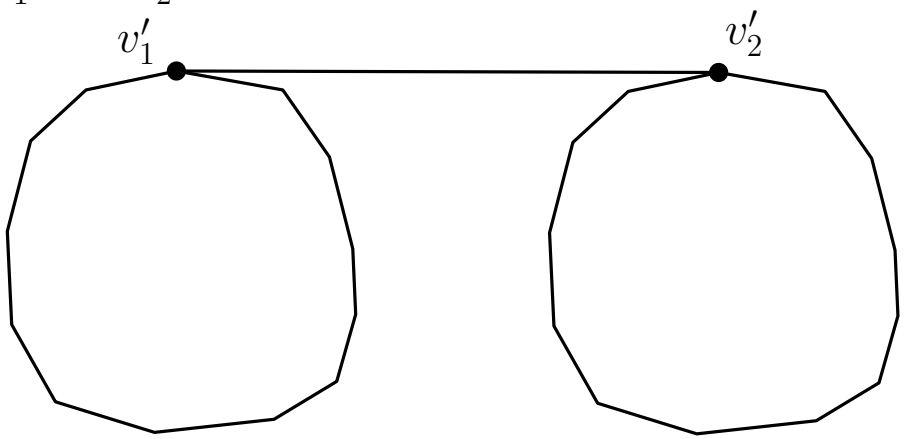
1. Orthogonally project C_1 and C_2 onto the plane $z = 0$. Let C_1' and C_2' respectively be their projections. The vertex v of greatest/smallest abscissa in C_i projects onto the vertex v' of greatest/smallest abscissa in C_i', so those are starting points to trace the boundaries of C_1' and C_2'
   - The following vertex of C_i' is the projection of one of the neigbors of v in C_i.
2. Find $v_1'v_2'$, one of the external bitangents of C_1' and C_2'.
3. Then v_1v_2 is an edge of $C \setminus C_1 \cup C_2$.



COMPUTING A CONVEX HULL IN 3D

Step 1: Finding the first new edge

1. Orthogonally project C_1 and C_2 onto the plane $z = 0$. Let C_1' and C_2' respectively be their projections. The vertex v of greatest/smallest abscissa in C_i projects onto the vertex v' of greatest/smallest abscissa in C_i', so those are starting points to trace the boundaries of C_1' and C_2'
   - The following vertex of C_i' is the projection of one of the neigbors of v in C_i.
2. Find $v_1'v_2'$, one of the external bitangents of C_1' and C_2'.
3. Then v_1v_2 is an edge of $C \setminus C_1 \cup C_2$.

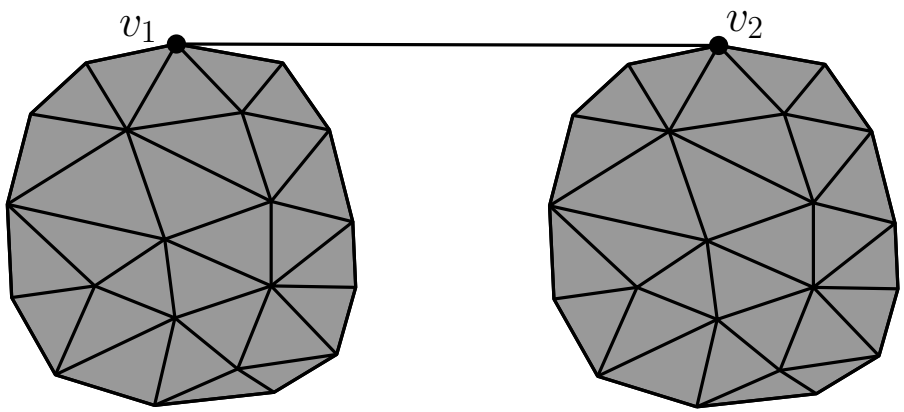


CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D

Step 1: Finding the first new edge

1. Orthogonally project C_1 and C_2 onto the plane $z = 0$. Let C_1' and C_2' respectively be their projections. The vertex v of greatest/smallest abscissa in C_i projects onto the vertex v' of greatest/smallest abscissa in C_i', so those are starting points to trace the boundaries of C_1' and C_2'
   - The following vertex of C_i' is the projection of one of the neigbors of v in C_i.
2. Find $v_1'v_2'$, one of the external bitangents of C_1' and C_2'.
3. Then v_1v_2 is an edge of $C \setminus C_1 \cup C_2$.

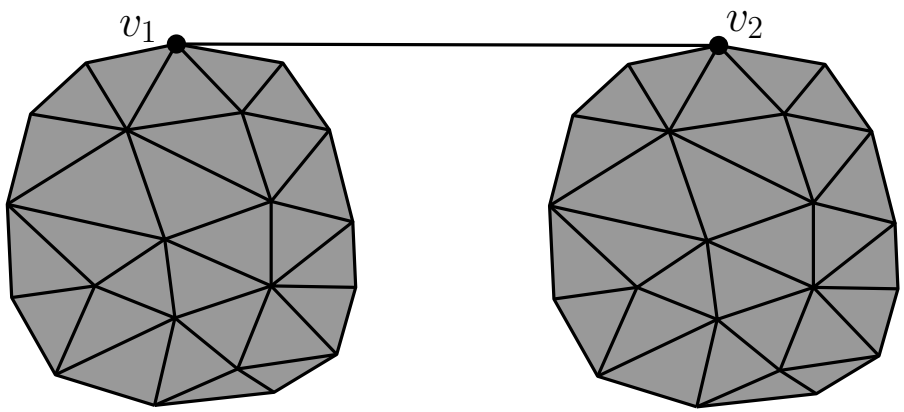


CONVEX HULLS IN 3D

COMPUTING A CONVEX HULL IN 3D

Step 1: Finding the first new edge

1. Orthogonally project C_1 and C_2 onto the plane $z = 0$. Let C_1' and C_2' respectively be their projections. The vertex v of greatest/smallest abscissa in C_i projects onto the vertex v' of greatest/smallest abscissa in C_i', so those are starting points to trace the boundaries of C_1' and C_2'
   - The following vertex of C_i' is the projection of one of the neigbors of v in C_i.
2. Find $v_1'v_2'$, one of the external bitangents of C_1' and C_2'.
3. Then v_1v_2 is an edge of $C \setminus C_1 \cup C_2$.



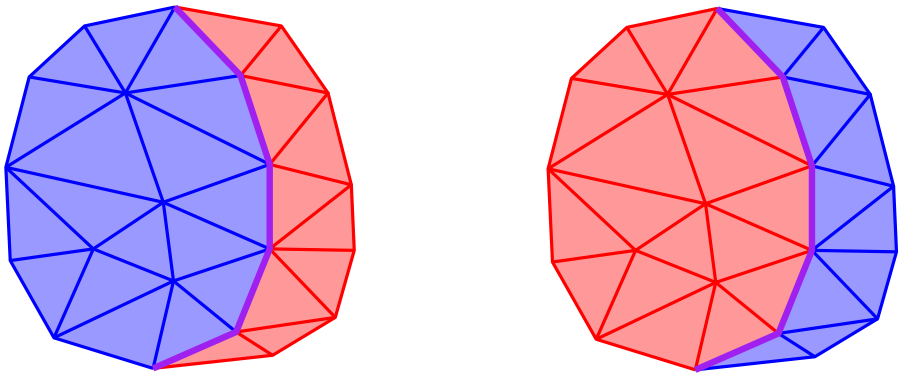
Running time: $O(n)$

1. Each edge of C_i is tested at most twice.
2. Common external tangents can be found in linear time.
3. Retrieving v_i from v_i' is done in constant time.

COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

We follow the *gift-wrapping* idea ⟶ *pivot a plane around the current new edge*

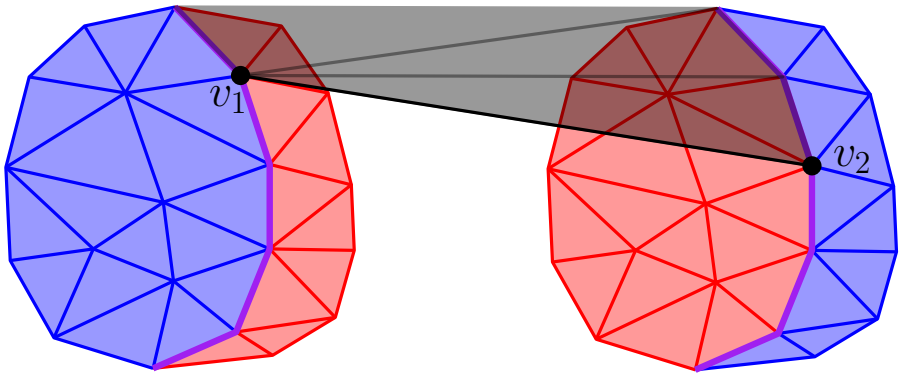


COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

We follow the *gift-wrapping* idea ⟶ *pivot a plane around the current new edge*

Let v_1v_2 be the last discovered new edge ($v_i \in C_i$).



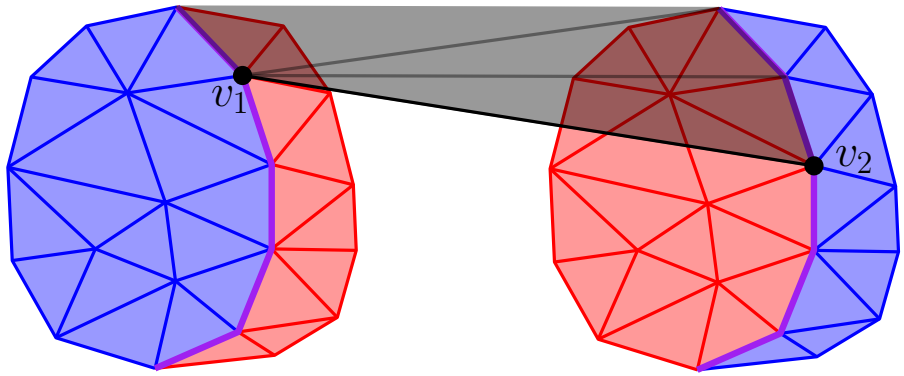
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

We follow the *gift-wrapping* idea ⟶ *pivot a plane around the current new edge*

Let v_1v_2 be the last discovered new edge ($v_i \in C_i$).

Let π be:
- If v_1v_2 is the first edge, π is the plane through v_1, v_2, v_1', v_2'.
- Otherwise, π is the plane containing the last discovered new triangle, which is incident to v_1v_2 on its left.



COMPUTING A CONVEX HULL IN 3D

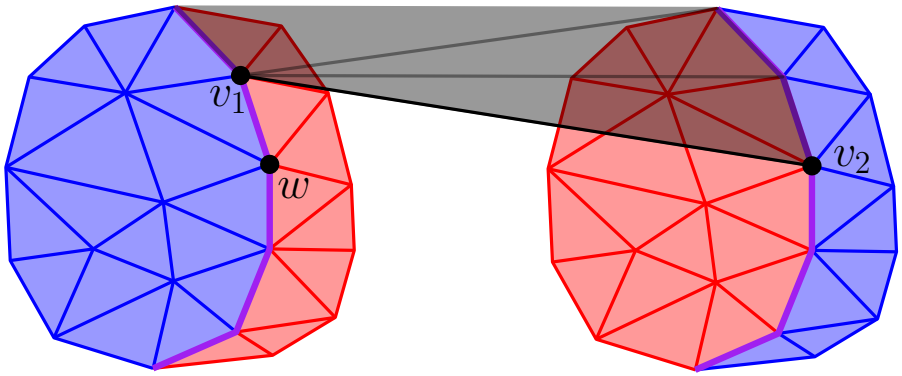
Step 2: Finding the remaining new faces and edges in the order induced by C_0

We follow the *gift-wrapping* idea ⟶ *pivot a plane around the current new edge*

Let v_1v_2 be the last discovered new edge ($v_i \in C_i$).

Let π be:
- If v_1v_2 is the first edge, π is the plane through v_1, v_2, v_1', v_2'.
- Otherwise, π is the plane containing the last discovered new triangle, which is incident to v_1v_2 on its left.

The next new face found is v_1v_2w. It is incident to v_1v_2 on its right, and w is the neighbor of either v_1 or v_2 forming smaller angle with π.



COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

Lemma 1

There is only one possible candidate $w_i \in C_i$ and it can be caracterized locally.

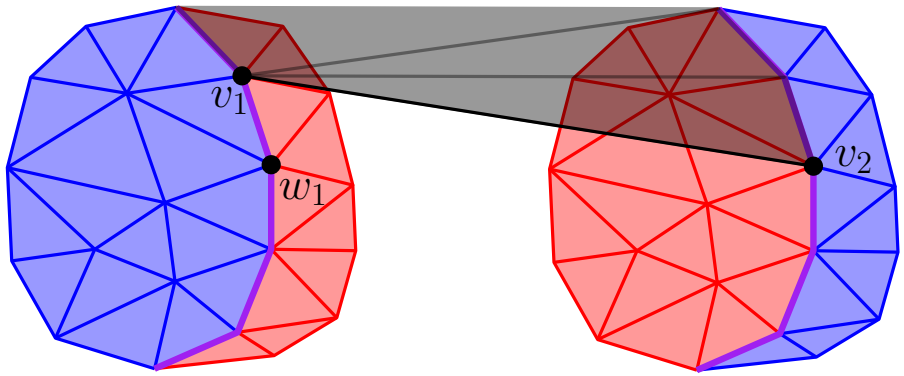
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

Lemma 1

There is only one possible candidate $w_i \in C_i$ and it can be caracterized locally.

Proof: W.l.o.g., let w_1 be a vertex adjacent to v_1. Let $p(w_1), s(w_1)$ respectively be its predecessor and its successor in the CCW circular order around the neighbors of v_1. Let h_1 be the plane $v_1w_1v_2$. Among the two halfspaces defined by h_1 let h_1^+ be the one opposite to vector $n = v_1w_1 \times v_1v_2$. If w_1 is a candidate, then h_1 supports C_1 and v_2. Therefore $p(w_1), s(w_1) \in h_1^+$. Reciprocally, if $p(w_1), s(w_1) \in h_1^+$ then $v_1w_1p(w_1)$ and $v_1w_1s(w_1)$ both support C_1. Therefore also h_1 supports C_1.



COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

Lemma 1

There is only one possible candidate $w_i \in C_i$ and it can be caracterized locally.

Proof: W.l.o.g., let w_1 be a vertex adjacent to v_1. Let $p(w_1), s(w_1)$ respectively be its predecessor and its successor in the CCW circular order around the neighbors of v_1. Let h_1 be the plane $v_1w_1v_2$. Among the two halfspaces defined by h_1 let h_1^+ be the one opposite to vector $n = v_1w_1 \times v_1v_2$. If w_1 is a candidate, then h_1 supports C_1 and v_2. Therefore $p(w_1), s(w_1) \in h_1^+$. Reciprocally, if $p(w_1), s(w_1) \in h_1^+$ then $v_1w_1p(w_1)$ and $v_1w_1s(w_1)$ both support C_1. Therefore also h_1 supports C_1.

Lemma 2

When v_i is incident to several new edges, the successive candidates are found in circular order about v_i.

COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

Lemma 1

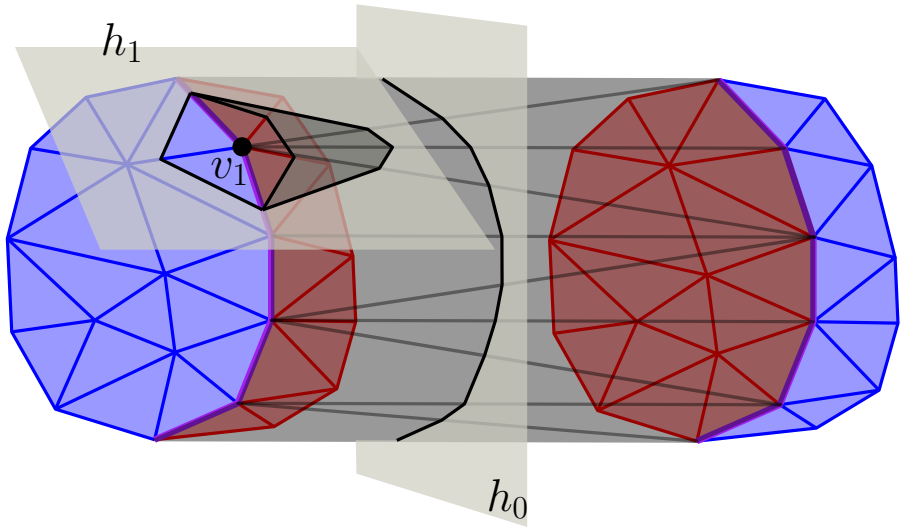
There is only one possible candidate $w_i \in C_i$ and it can be caracterized locally.

Proof: W.l.o.g., let w_1 be a vertex adjacent to v_1. Let $p(w_1), s(w_1)$ respectively be its predecessor and its successor in the CCW circular order around the neighbors of v_1. Let h_1 be the plane $v_1w_1v_2$. Among the two halfspaces defined by h_1 let h_1^+ be the one opposite to vector $n = v_1w_1 \times v_1v_2$. If w_1 is a candidate, then h_1 supports C_1 and v_2. Therefore $p(w_1), s(w_1) \in h_1^+$. Reciprocally, if $p(w_1), s(w_1) \in h_1^+$ then $v_1w_1p(w_1)$ and $v_1w_1s(w_1)$ both support C_1. Therefore also h_1 supports C_1.

Lemma 2

When v_i is incident to several new edges, the successive candidates are found in circular order about v_i.

Proof: WLG, v_1 can be separated from the remaining vertices of C_1 and C_2 by a plane h_1, which intersects all the edges of C_1 and C incident to v_1 forming two polytopes. The circular order of the vertices of $C_1 \cap h_1$ and $C \cap h_1$ is the same, and it also coincides with the circular order of $C \cap h_0$.



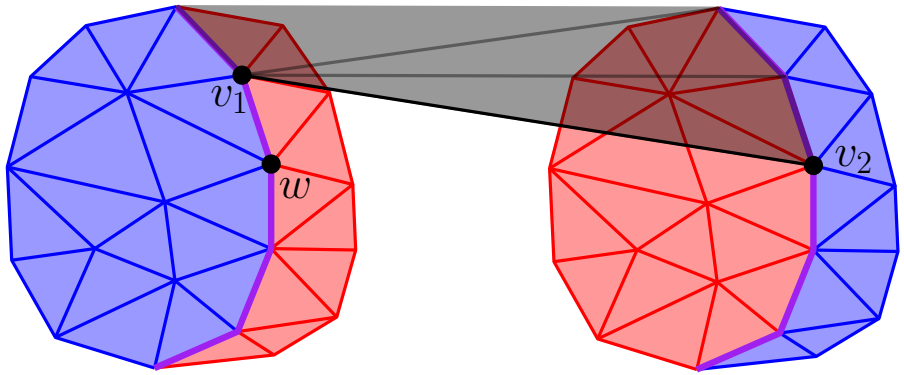
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

At each step:

1. From face uv_1v_2 find the best candidates w_1, adjacent to v_1 and w_2 adjacent to v_2.
2. Choose w to be the best of w_1 and w_2.
3. You have found:
   - a purple vertex w
   - a purple edge v_iw
   - a new triangle v_1v_2w

   that can be added to the resulting convex hull



CONVEX HULLS IN 3D

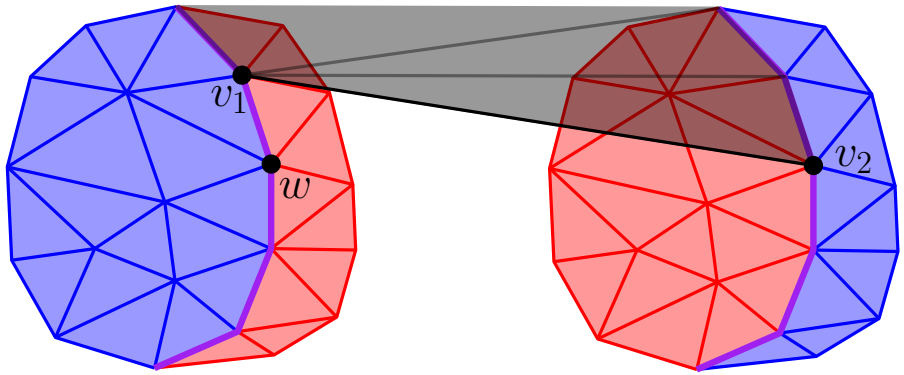
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

At each step:

1. From face uv_1v_2 find the best candidates w_1, adjacent to v_1 and w_2 adjacent to v_2.
2. Choose w to be the best of w_1 and w_2.
3. You have found:
   - a purple vertex w
   - a purple edge v_iw
   - a new triangle v_1v_2w

   that can be added to the resulting convex hull



Running time: $O(n)$

At each step, the neighbors of v_i are tested in order (Lemma 2) and each test takes $O(1)$ time (Lemma 1).

CONVEX HULLS IN 3D

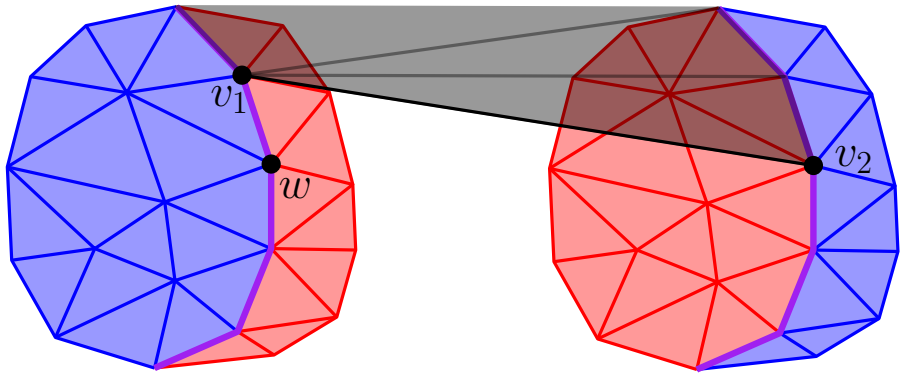
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

At each step:

1. From face uv_1v_2 find the best candidates w_1, adjacent to v_1 and w_2 adjacent to v_2.
2. Choose w to be the best of w_1 and w_2.
3. You have found:
   - a purple vertex w
   - a purple edge v_iw
   - a new triangle v_1v_2w

   that can be added to the resulting convex hull



Running time: $O(n)$

At each step, the neighbors of v_i are tested in order (Lemma 2) and each test takes $O(1)$ time (Lemma 1).

Step 3: Identify the red faces, edges and vertices, and update the convex hull

CONVEX HULLS IN 3D

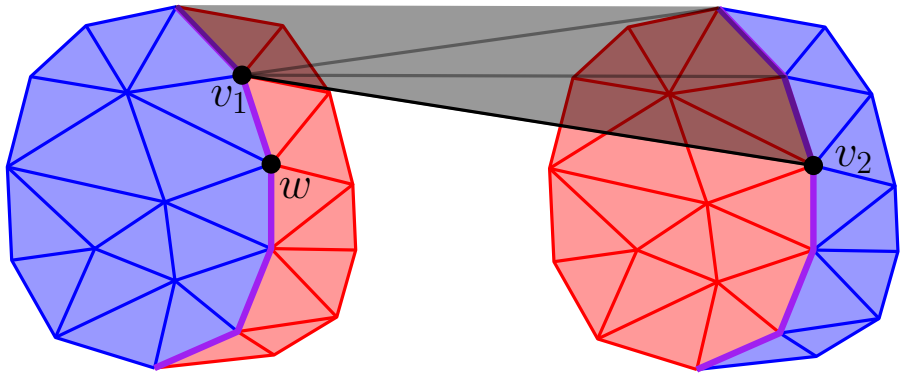
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

At each step:

1. From face uv_1v_2 find the best candidates w_1, adjacent to v_1 and w_2 adjacent to v_2.
2. Choose w to be the best of w_1 and w_2.
3. You have found:
   - a purple vertex w
   - a purple edge v_iw
   - a new triangle v_1v_2w

   that can be added to the resulting convex hull



Running time: $O(n)$

At each step, the neighbors of v_i are tested in order (Lemma 2) and each test takes $O(1)$ time (Lemma 1).

Step 3: Identify the red faces, edges and vertices, and update the convex hull

In the previous step, each time a purple edge is found, its incident faces are labeled blue or red.

CONVEX HULLS IN 3D

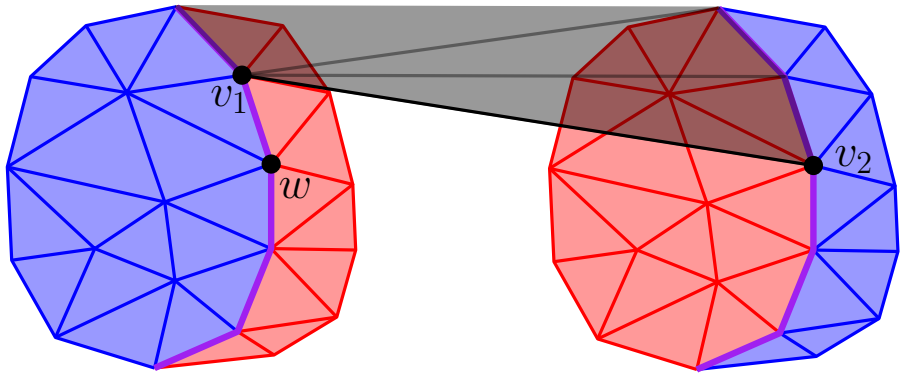
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

At each step:

1. From face uv_1v_2 find the best candidates w_1, adjacent to v_1 and w_2 adjacent to v_2.
2. Choose w to be the best of w_1 and w_2.
3. You have found:
   - a purple vertex w
   - a purple edge v_iw
   - a new triangle v_1v_2w

   that can be added to the resulting convex hull



Running time: $O(n)$

At each step, the neighbors of v_i are tested in order (Lemma 2) and each test takes $O(1)$ time (Lemma 1).

Step 3: Identify the red faces, edges and vertices, and update the convex hull

In the previous step, each time a purple edge is found, its incident faces are labeled blue or red.

In this step:

1. All unlabelled faces adjacent to a red face are recursively labelled red.
2. All unlabelled edges incident to a red face are labelled red.
3. All unlabelled vertex incident to a red or purple edge is labelled red.
4. Remove all red faces, edges and vertices from (the representation of) the convex hull.

CONVEX HULLS IN 3D

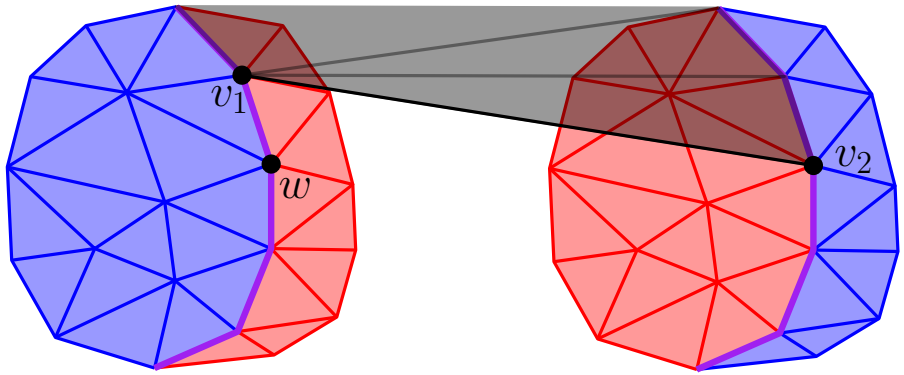
COMPUTING A CONVEX HULL IN 3D

Step 2: Finding the remaining new faces and edges in the order induced by C_0

At each step:

1. From face uv_1v_2 find the best candidates w_1, adjacent to v_1 and w_2 adjacent to v_2.
2. Choose w to be the best of w_1 and w_2.
3. You have found:
   - a purple vertex w
   - a purple edge v_iw
   - a new triangle v_1v_2w

   that can be added to the resulting convex hull



Running time: $O(n)$

At each step, the neighbors of v_i are tested in order (Lemma 2) and each test takes $O(1)$ time (Lemma 1).

Step 3: Identify the red faces, edges and vertices, and update the convex hull

In the previous step, each time a purple edge is found, its incident faces are labeled blue or red.

In this step:

1. All unlabelled faces adjacent to a red face are recursively labelled red.
2. All unlabelled edges incident to a red face are labelled red.
3. All unlabelled vertex incident to a red or purple edge is labelled red.
4. Remove all red faces, edges and vertices from (the representation of) the convex hull.

Running time: $O(n)$

COMPUTING A CONVEX HULL IN 3D

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.

COMPUTING A CONVEX HULL IN 3D

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.
   1. Find an edge of $C \setminus C_1 \cup C_2$
   2. Find the remaining new faces and edges in the order induced by C_0
   3. Identify the red faces, edges and vertices and update the convex hull

COMPUTING A CONVEX HULL IN 3D

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.
   1. Find an edge of $C \setminus C_1 \cup C_2$
   2. Find the remaining new faces and edges in the order induced by C_0
   3. Identify the red faces, edges and vertices and update the convex hull

$O(n)$ time

COMPUTING A CONVEX HULL IN 3D

Algorithm

Input: $p_1, \ldots, p_n \in \mathbb{R}^3$
Output: $ch(p_1, \ldots, p_n)$

1. Initialization
   Sort $p_1, \ldots, p_n$ by abscissa.
2. Division
   Partition set $P = \{p_1, \ldots, p_n\}$ into two equally sized subsets P_1 and P_2 by means of a vertical plane h_0.
3. Recursion
   Compute $C_1 = ch(P_1)$ and $C_2 = ch(P_2)$.
4. Merging
   Compute $C = ch(C_1 \cup C_2)$.
   1. Find an edge of $C \setminus C_1 \cup C_2$
   2. Find the remaining new faces and edges in the order induced by C_0
   3. Identify the red faces, edges and vertices and update the convex hull

$O(n)$ time

Theorem: The algorithm computes the convex hull of n points in E^3 in $O(n \log n)$ time and $O(n)$ space. These bounds are optimal.

FURTHER READING

J.-D. Boissonat. M. Yvinec, **Algorithmic Geometry**, Cambridge University Press, 1998.